Data Cleaning, Integration, and Analysis Part 2

We will perform a comprehensive upgrade and organization of the existing data in Excel to facilitate future database construction. Some parts of the code need to be filled in.

Part 1: Groupby

The file_path is used to read the Excel file. Then, find the sheet_name of the Excel file and name the entire large table as `df`. The df[…] calls refer to each column of the large table.

```
import pandas as pd

file_path = 'C:/Users/Frank/Desktop/DHT11 Data Collection.xlsx'  # Define the
path to the Excel file
xls = pd.ExcelFile(file_path)  # Load the Excel file into a Pandas ExcelFile
object

sheet_names = xls.sheet_names  # Get the sheet names in the Excel file

df = pd.read_excel(file_path, sheet_name='Sheet1')  # Load the first sheet in the
Excel file into a DataFrame

df['时间 (Timestamp) '] = pd.to_datetime(df['时间 (Timestamp) '])  # Convert the
'Timestamp' column to datetime format

# Group by 'TimeZone' and calculate the mean and standard deviation of
'Temperature' and 'Humidity'
grouped = df.groupby('早/中/下午/晚上/夜间 (TimeZone) ').agg({
    '温度 (Temperature) ': ['mean', 'std'],
    '湿度 (Humidity) ': ['mean', 'std']
}).reset_index()

# Flatten MultiIndex columns and rename them for clarity
grouped.columns = ['时间段 (TimeZone) ', '温度均值 (Mean Temperature) ', '温度标准
差 (Temperature Std Dev) ',
                   '湿度均值 (Mean Humidity) ', '湿度标准差 (Humidity Std Dev) ']

# Calculate the rate of change for 'Temperature' and 'Humidity' and convert to
percentage
df['温度变化率 (Temperature Change Rate) '] = df['温度 (Temperature)
'].pct_change() * 100
```

```
df['湿度变化率（Humidity Change Rate）'] = df['湿度（Humidity）'].pct_change() *
100

# Calculate the Comfort Index using the formula: Temperature - Humidity * 0.55 +
(Temperature - Mean Temperature) * 0.1
df['舒适指数（Comfort Index）'] = df['温度（Temperature）'] - df['湿度（Humidity）
'] * 0.55 + (df['温度（Temperature）'] - df['温度（Temperature）'].mean()) * 0.1

# Replace 0 values in the temperature change rate with 'no change'
df['温度变化率（Temperature Change Rate）'] = df['温度变化率（Temperature Change
Rate）'].apply(
    lambda x: 'no change' if x == 0 else x
)

# Replace 0 values in the humidity change rate with 'no change'
df['湿度变化率（Humidity Change Rate）'] = df['湿度变化率（Humidity Change Rate）
'].apply(
    lambda x: 'no change' if x == 0 else x
)

df = df.drop(columns=['信号强度（Signal Strength）', '数据来源（Data Source）', '备
注（Remarks）'])  # Drop unnecessary columns

# Rename columns
df = df.rename(columns={
    'ID（唯一标识符）': 'ID',
    '地点（Location）': 'Location',
    '温度（Temperature）': 'Temperature',
    '湿度（Humidity）': 'Humidity',
    '时间（Timestamp）': 'Timestamp',
    '传感器 ID（Sensor ID）': 'Sensor ID',
    '早/中/下午/晚上/夜间（TimeZone）': 'TimeZone',
    '温度变化率（Temperature Change Rate）': 'Temperature Change Rate',
    '舒适指数（Comfort Index）': 'Comfort Index',
    '湿度变化率（Humidity Change Rate）': 'Humidity Change Rate'
})

output_file_path_final =
'C:/Users/Frank/Desktop/Updated_DHT11_Data_Collection_v1.xlsx'  # Define the path
to save the updated Excel file
```

```
df.to_excel(output_file_path_final, index=False)  # Save the DataFrame to an
Excel file without the index
print(f"Data has been saved to {output_file_path_final}")  # Print the path of
the saved file
```

```
# Group by 'TimeZone' and calculate the mean and standard deviation of
'Temperature' and 'Humidity'
grouped = df.groupby('早/中/下午/晚上/夜间 (TimeZone) ').agg({
    '温度 (Temperature) ': ['mean', 'std'],
    '湿度 (Humidity) ': ['mean', 'std']
}).reset_index()
```

The df.groupby syntax is new and means that we group by the 'TimeZone' column, and then calculate the mean and standard deviation for these groups.

df.groupby('早/中/下午/晚上/夜间（TimeZone）'):

- **Groups the data by the 'TimeZone' column.** Each group represents a different time of the day, such as morning, noon, afternoon, evening, and night.

.agg({ '温度（Temperature）': ['mean', 'std'], '湿度（Humidity）': ['mean', 'std'] }):

- **Uses the agg function to calculate specified statistics for each group.**
- Computes the mean and standard deviation for 'Temperature' and 'Humidity'.

.reset_index():

- **Resets the index of the grouped result, making the group keys (time periods) regular columns in the DataFrame.**

```
# Flatten MultiIndex columns and rename them for clarity
grouped.columns = ['时间段 (TimeZone) ', '温度均值 (Mean Temperature) ', '温度标准
差 (Temperature Std Dev) ',
                   '湿度均值 (Mean Humidity) ', '湿度标准差 (Humidity Std Dev) ']
```

After groupby and agg operations, the generated DataFrame column names will become multi-layer indexes (MultiIndex), such as ('Temperature', 'mean') and ('Temperature', 'std').

Flattening: **Convert multi-layer index column names to single-layer indexes, which is more concise and easier to handle.**

```
# Calculate the rate of change for 'Temperature' and 'Humidity' and convert to
percentage
df['温度变化率 (Temperature Change Rate) '] = df['温度 (Temperature)
'].pct_change() * 100
df['湿度变化率 (Humidity Change Rate) '] = df['湿度 (Humidity) '].pct_change() *
100
```

Explanation of Calculating Rate of Change:

Temperature Change Rate: .pct_change() function is straightforward; **it compares the temperature at the current time with the temperature at the previous time.**
The .pct_change() function in Pandas calculates the percentage change between consecutive data points. It computes the relative change between each data point and the previous one.

Percentage Change Rate= (Current Value−Previous Value) / Previous Value ×100%

```
# Replace 0 values in the temperature change rate with 'no change'
df['温度变化率 (Temperature Change Rate) '] = df['温度变化率 (Temperature Change
Rate) '].apply(
    lambda x: 'no change' if x == 0 else x
)

# Replace 0 values in the humidity change rate with 'no change'
df['湿度变化率 (Humidity Change Rate) '] = df['湿度变化率 (Humidity Change Rate)
'].apply(
    lambda x: 'no change' if x == 0 else x
)
```

These two code segments replace the values of 0 in the temperature and humidity change rates with the string "no change".

- apply function: Applies a function to each value in a specified column of the DataFrame.
- lambda function: An anonymous function used to define simple processing logic. It checks if each value is 0.
- if condition: Checks if each value is 0; if true, it replaces the value with "no change"; otherwise, it retains the original value.